Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Rok Černič

Prototip informacijskega sistema za podporo študentskemu klubu

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Damjan Vavpotič

Ljubljana 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V okviru diplomske naloge razvijte delujoč prototip sistema za podporo študentskemu klubu. Funkcionalnosti prototipa naj temeljijo na predhodni analizi obstoječih podobnih sistemov ter na analizi dejanskih potreb članov študentskega kluba. Ključne rezultate te analize v delu predstavite in na podlagi le-teh načrtujte in izdelajte delujoč prototip, ki bo vključeval tako strežniški kot odjemalski del in v smiselnem obsegu deloval tudi na mobilnih napravah. V diplomski nalogi predstavite ključne funkcionalnosti, programsko arhitekturo in delovanje prototipa. Pripravljen prototip kritično ovrednotite in podajte smernice za nadaljnje delo.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Rok Černič sem avtor diplomskega dela z naslovom:

Prototip informacijskega sistema za podporo študentskemu klubu

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Damjana Vavpotiča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 1. januarja 2016 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Damjanu Vavpotiču za pomoč in vodenje pri izdelovanju diplomskega dela. Prav tako se zahvaljujem punci, družini in prijateljem za podporo pri študiju in diplomskem delu.

Pokojni mami Alenki.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	Application Programming Interface	programski vmesnik
CSS	Cascading style sheet	prekrivni slogi
HHVM	HipHop Virtual Machine	navidezni stroj HipHop
HTML	HyperText Markup Language	označevalni jezik
JSON	JavaScript Object Notation	notacija objektov JavaScript
KBŠ		Klub belokranjskh študentov
MVC	Model-View-Controller	model-pogled-krmilnik
PHP	Hypertext Preprocessor	skriptni programski jezik
SASS	Syntactically Awesome Style Sheets	predprocesorski jezik CSS-ja
SQL	Structured Query Language	strukturirani povpraševalni jezik
VCS	Version control systems	sistem za nadzor različic kode

Povzetek

V sklopu diplomske naloge smo razvili spletno in mobilno aplikacijo za člane Kluba belokranjskih študentov. Uporabili smo aktualne tehnologije na področju razvoja mobilnih in spletnih aplikacij. Naredili smo analizo potreb bodočih uporabnikov in obstoječih informacijskih sistemov za podporo klubov. V delu smo predstavili vse ključne korake razvoja obeh aplikacij. Spletna aplikacija je namenjena članom kluba in njihovem boljšem obveščanju. Omogoča tudi pregled aktualnih dogodkov, novic, prevozov itd. Mobilna aplikacija je namenjena upravljanju in pregledu prevozov.

Ključne besede: spletna aplikacija, mobilna aplikacija, centralizirano upravaljanje, prevozi, Klub belokranjskih študentov, Likertova lestvica, Laravel, Ionic, analiza uporabnikov.

Abstract

We have developed a web and mobile application for student club members. We have used the latest technology in the field of development of mobile and web applications. We have made an analysis of the needs of future users and analysis of existing information systems. In addition, we also presented the steps of developing of both applications. Web application is designed for club members and their better informing. It also allows review of current events, news, services etc. Mobile application is designed to manage and review transports.

Keywords: web application, mobile application, centralized management, students club, Likert scale, Laravel, Ionic, analysis of users.

Poglavje 1

Uvod

Klub belokranjskih študentov je največja mladinska organizacija, ki združuje mlade, predvsem študente in dijake iz vseh treh belokranjskih občin: Črnomelj, Metlika in Semič. Klub je bil formalno ustanovljen 15. januarja 1963, ko se je odcepil od Kluba dolenjskih študentov. Njegovo delovanje je že od takrat naprej usmerjeno predvsem v kreativno aktivnost mladih.

Zaradi aktualne spletne strani, ki je bila izdelana v letu 2008 je vodstvo kluba prišlo do zaključka, da je treba spletno stran popolnoma prenoviti. Zaradi porasta uporabe mobilnih tehnologij se je porodila tudi potreba po mobilni aplikaciji.

Prenovljen sistem omogoča boljšo povezavo z aktualnimi socialnimi omrežji in njegovo vodenje. Pojavila se je potreba po možnosti registracije uporabnikov, saj to omogoča razlikovanje uporabnikov in boljše naslavljanje posameznega uporabnika z določenimi informacijami.

Diplomska naloga je razdeljena na osem poglavij. V drugem poglavju smo analizirali obstoječe informacijske sisteme za podporo klubov po Sloveniji. S pomočjo te analize smo dobili skupek funkcionalnosti, ki smo jih vključili v anketo v tretjem poglavju, kjer smo izvedli analizo bodočih uporabnikov. Iz te smo dobili funkcionalnosti, ki so uporabnikom najpomembnejše, in smo jih nato tudi implementirali v prototip. V četrtem poglavju smo na kratko opisali tehnologije in orodja, uporabljena pri razvoju. V petem poglavju

smo opisali arhitekturo sistema. V šestem smo pokazali razvoj prototipa in v sedmem predstavili funkcionalnosti. V osmem smo obravnavali možnosti nadgradnje.

Poglavje 2

Analiza obstoječih informacijskih sistemov

V času pisanja diplomskega dela je bilo v Sloveniji registriranih 56 študentskih klubov, od teh jih je 47 imelo delujočo spletno stran.

Večina spletnih aplikacij drugih klubov je narejena na podoben način. Na prvi strani prikažejo novice in prihajajoče dogodke. Večina jih vsebuje tudi galerijo, ki vsebuje slike iz preteklih dogodkov. Pri vseh lahko najdemo kontaktne podatke, vendar jih ima samo peščica integriran kontaktni obrazec v spletni aplikaciji. Večina jih ima naštete tudi ugodnosti za študente. Vsebujejo tudi navodila in PDF-obrazec za včlanitev v klub ali podaljšanje članstva. Vendar jih le nekaj omogoča sklenitev in podaljšanje članstva elektronsko. Tabela 2.1 prikazuje primerjavo dvanajst naključnih obstoječih sistemov klubov.

Klub	Fukcionalnost										
	1	2	3	4	5	6	7	8	9	10	11
A	Da	Da	Ne	Ne	Da	Ne	Da	Ne	Ne	Da	Ne
B	Ne	Da	Ne	Ne	Da	Da	Da	Ne	Ne	Da	Ne
C	Da	Da	Ne	Ne	Da	Da	Ne	Ne	Ne	Ne	Ne
D	Da	Da	Ne	Ne	Da	Da	Da	Ne	Ne	Ne	Ne
E	Ne	Da	Ne	Ne	Da	Ne	Da	Ne	Ne	Da	Ne
F	Da	Da	Ne	Ne	Da	Da	Da	Ne	Ne	Da	Ne
G	Ne	Da	Da	Da	Da	Ne	Ne	Ne	Ne	Da	Ne
H	Ne	Da	Ne	Ne	Da	Da	Da	Ne	Ne	Ne	Ne
I	Ne	Da	Ne	Ne	Da	Da	Da	Ne	Ne	Da	Ne
J	Da	Da	Ne	Ne	Da	Da	Da	Ne	Ne	Ne	Ne
K	Ne	Da	Ne	Ne	Ne	Da	Da	Ne	Ne	Da	Ne
L	Da	Da	Ne	Ne	Da	Ne	Da	Ne	Ne	Da	Ne
M	Ne	Da	Ne	Ne	Ne	Ne	Da	Ne	Ne	Da	Ne
N	Da	Da	Ne	Ne	Da	Da	Da	Ne	Ne	Da	Ne
O	Ne	Da	Ne	Ne	Da	Da	Ne	Ne	Ne	Ne	Ne
P	Ne	Da	Ne	Ne	Da	Na	Da	Ne	Ne	Da	Ne
R	Ne	Da	Ne	Ne	Da	Da	Da	Ne	Ne	Ne	Ne
S	Da	Da	Ne	Ne	Da	Ne	Ne	Ne	Ne	Ne	Ne
T	Ne	Da	Da	Da	Da	Da	Da	Ne	Ne	Ne	Ne

A: Društvo novomeških študentov
B: Društvo študentov Brežice
C: Klub ajdovskih študentov in dijakov
D: Klub goriških študentov
E: Klub idrijskih študentov
F: Klub kočevskih študentov
G: Klub koroških študentov
H: Klub litijskih in šmarskih študentov
I: Klub mariborskih študentov
J: Klub ormoških študentov
K: Klub posavskih študentov
L: Klub prekmurskih študentov
M: Klub ptujskih študentov
N: Klub škofjeloških študentov
O: Klub študentov Ilirska Bistrica
P: Klub študentov Kranj
R: Klub študentov občin Postojna in Pivka
S: Klub študentov občine Piran
T: Klub radovljiških študentov

1: kontaktni obrazec
2: pregled novic
3: včlanitev v klub
4: podaljšanje članstva
5: pregled ugodnosti za študente
6: galerija
7: pregled dogodkov
8: prijava na dogodke
9: prodaja vstopnic
10: pregled projektov
11: prijava na projekt

Tabela 2.1: Primerjava spletnih aplikacij drugih klubov.

Poglavje 3

Analiza potreb bodočih uporabnikov

Namen analize je bil pridobiti najpomembnejše funkcionalnosti bodočemu uporabniku spletne aplikacije. Te funkcionalnosti smo nato implementirali v prototip informacijskega sistema.

3.1 Opis populacije in vzorca

V našo populacijo so spadali vsi aktivni člani kluba. Torej člani, ki redno spremljajo spletno aplikacijo kluba in jih je skupaj približno 150. V vzorec je bilo vključeno 29 moških in 29 ženskih anketirancev.

Vprašalnik smo razširili med anketirance tako, da smo v obstoječi spletni aplikaciji in v Facebook skupino objavili povezavo do ankete (Slika 3.1).

3.2 Vprašalnik

Podatke smo zbrali z anonimnim anketnim vprašalnikom. S prvim vprašanjem smo izvedeli spol anketiranca. Preostali del pa so sestavljale trditve v povezavi z opisi posameznih funkcionalnostih. Študenti so svoje strinjanje ocenjevali na 7-stopenjski Likertovi lestvici. Ocena 1 je pomenila, da se sploh ne

Slika 3.1: Objava v Facebook skupini.

strinjajo, ocena 7 pa, da se popolnoma strinjajo [1]. Vprašalnik je vseboval 11 trditev, ki so bile sestavljene na podoben način. Pred samo trditvijo je bil opis, ki je anketirancu na kratko predstavil funkcionalnost, na katero se je trditev nanašala.

`Obrazce, s katerim lahko kontaktirate vodstvo kluba in oddate svoje pritožbe, pohvale in predloge. Ta funkcionalnost je zelo pomembna v okviru spletne aplikacije za KBŠ.`

3.3 Obdelava podatkov

Podatke smo obdelali s pomočjo programa SPSS. Izračunali smo povprečno oceno strinjanja o potrebnosti posamezne funkcionalnosti, standardni odklon, frekvenco in odstotek za posamezno trditev [2]. Razlike med spoloma smo ugotavljali z analizo variance.

	$\bar{x}$	δ	x_{min}	x_{max}
kontakt	6,00	1,043	3	7
pregled novic	6,14	1,330	1	7
včlanitev v klub	5,91	1,548	1	7
podaljšanje članstva	5,71	1,578	1	7
pregled ugodnosti za študente	5,86	1,249	2	7
galerija	6,07	0,856	4	7
pregled dogodkov	6,50	0,884	2	7
prijava na dogodke	5,84	1,335	1	7
prodaja vstopnic	5,22	1,633	1	7
pregled projektov	5,90	0,949	4	7
prijava na projekt	5,40	1,509	1	7

$\bar{x}$ = povprečje rezultatov
δ = standardni odklon
x_{min} = najnižji rezultat
x_{max} = najvišji rezultat

Tabela 3.1: Funkcionalnosti: povprečje rezulatov in standardni odklon.

3.4 Rezultati

3.4.1 Funkcionalnosti za implementacijo v prototip

Tabela 3.1 nam prikazuje povprečje vsake trditve, standardni odklon ter najnižji in najvišji rezultat.

Tabela 3.2 nam prikazuje frekvence in odstotke za posamezen odgovor pri posamezni trditvi. Člani so pri večini trditvah odgovorili, da se popolnoma strinjajo s funkcionalnostjo.

Pri izdelovanju prototipa smo implementirali prvih 6 funkcionalnosti z največjim povprečjem v tabeli 3.1.

	frekvence (F)							odstotki (%)						
	1	2	3	4	5	6	7	1	2	3	4	5	6	7
A	0	0	2	3	10	21	22	0,0	0,0	3,4	5,2	17,2	36,2	**37,9**
B	2	0	0	4	5	16	31	3,4	0,0	0,0	6,9	8,6	27,6	**53,4**
C	3	1	0	3	7	17	27	5,2	1,7	0,0	5,2	12,1	29,3	**46,6**
D	2	0	2	12	4	11	27	3,4	0,0	3,4	20,7	6,9	19,0	**46,6**
E	0	1	2	5	12	14	24	0,0	1,7	3,4	8,6	20,7	24,1	**41,4**
F	0	0	0	2	13	22	21	0,0	0,0	0,0	3,4	22,4	**37,9**	36,2
G	0	1	0	0	5	14	38	0,0	1,7	0,0	0,0	8,6	24,1	**65,5**
H	1	1	1	3	16	11	25	1,7	1,7	1,7	5,2	27,6	19,0	**43,1**
I	1	3	5	10	11	10	18	1,7	5,2	8,6	17,2	19,0	17,2	**31,0**
J	0	0	0	5	14	21	18	0,0	0,0	0,0	8,6	24,1	**36,2**	31,0
K	2	0	4	10	9	17	16	3,4	0,0	6,9	17,2	15,5	**29,3**	27,6

A: kontakt

B: pregled novic

C: včlanitev v klub

D: podaljšanje članstva

E: pregled ugodnosti za študente

F: galerija

G: pregled dogodkov

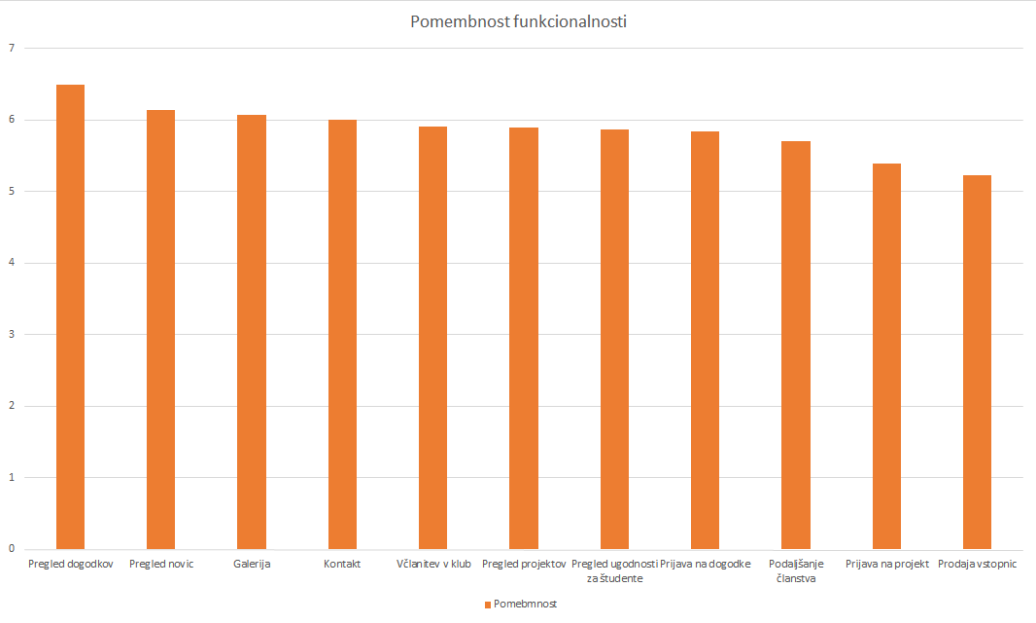
H: prijava na dogodke

I: prodaja vstopnic

J: pregled projektov

K: prijava na projekt

Tabela 3.2: Funkcionalnosti: frekvence in odstotki za posamezno trditev.



Slika 3.2: Pomembnost funkcionalnosti glede na povprečne vrednosti odgovorov.

3.4.2 Primerjava funkcionalnosti glede na spol

Razlike med spoloma smo ugotavljali z analizo variance ANOVA, ki velja za zelo robustno metodo. Zahteva določene pogoje, ki morajo biti izpolnjeni, da je test zanesljiv in verodostojen. Prvi pogoj je, da so podatki porazdeljeni normalno, drugi pa homogenost varianc. To pomeni, da morajo biti variance populacij za vsako skupino enake oziroma med njimi ne sme obstajati statistično pomembna razlika. [3]

V analizi sta oba pogoja izpolnjena. Rezultat analize pa je pokazal, da med spoloma ni statističnih razlik glede funkcionalnosti.

Poglavje 4

Izbor tehnologij in orodij za razvoj

Pri izdelavi prototipa smo uporabili različne vrste, že napisanih knjižnic in orodij, ki so nam pomagali pri izdelavi spletnih aplikacij. V nadaljevanju jih bomo opisali.

4.1 Laravel

Laravel je eno od najpopularnejših prosto dostopnih ogrodij, napisanih v PHP-jeziku [4]. Zagotavlja uveljavljene principe in arhitekturo za pomoč pri izdelavi spletnih strani. Z uporabo ogrodja si zagotovimo razbremenitev dela pri programiranju preprostih nalog, s katerimi se srečamo.

Primeri nalog, na katere naletimo pri izdelavi spletne aplikacije:

- obvladovanje sej,
- predpolnjenje,
- avtentikacija,
- obvladovanje piškotkov,
- usmerjanje,

- varnost.

Vse spletne aplikacije vsebujejo sejo, v katerih hranijo podatke, ki jih uporabljajo pri delovanju. Laravel nam tukaj pomaga tako, da ima že v naprej pripravljene ukaze, s katerimi na hitro nekaj zapišemo ali preberemo iz seje. Predpolnjenje uporabljamo, kadar imamo na voljo podatke, ki jih aplikacija pogosto potrebuje. Te podatke shranimo v predpomnilnik zato, da zmanjšamo število zahtevkov na podatkovno bazo. Posledično tudi skrajšamo čas dostopa do njih. Avtentikacijo uporabljamo pri spletnih straneh, ki vsebujejo neke vrste prijave oz. registracije uporabnika. Laravel nam pri avtentikaciji že v naprej pripravi potrebne poglede (view) in kontrolerje za postavitev prijave. Poleg tega poskrbi tudi za varnost. Piškotke uporabimo v primeru kadar, želimo za daljši čas shraniti določene podatke v brskalnik. Usmerjanje se uporablja pri manipulaciji URL-naslovov v projektu, najpomembnejši del spletne aplikacije pa je varnost. Laravel za osnovno varnost poskrbi brez dodatne kode in truda razvijalca. [5]

Za izdelavo prototipa smo uporabili Laravel 5.0. Za uporabo te verzije potrebujemo PHP-različico vsaj 5.4. Poleg tega smo uporabili še Mcrypt, OpenSSL, Mbstring in razširitev Tokenizer PHP.

4.2 Ionic

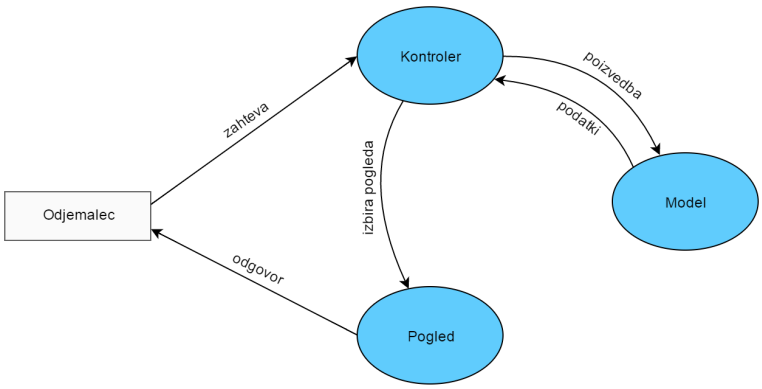
Ionic je ogrodje za izdelavo mobilnih aplikacij s pomočjo spletnih tehnologij. Zgrajeno je na temeljih popularnega ogrodja z imenom Cordova. Omogoča izdelavo aplikacije z jeziki, kot so HTML, CSS in JavaScript. Najprivlačnejše pri tem ogrodju pa je to, da pri trenutni različici lahko iz ene skupne kode zgradimo aplikacijo, ki teče na telefonih z Androidom in poleg tega lahko zgradimo aplikacijo za sisteme IOS. Ionic v ozadju uporablja popularno priljubljeno JavaScript ogrodje Angular, ki v projekt uvede dobre prakse, s katerimi zagotovimo lažje branje kode in kasnejše nadgrajevanje aplikacije. [6]

4.3 Composer in Bower

Composer je PHP-orodje, ki nam olajša namestitev različnih knjižnic, ki jih uporabljamo v projektu. Poleg tega nam pomaga tudi pri nadgradnji knjižnic. Z enim ukazom zaženemo iskanje novejših različic in nato tudi namestitev. Bower je podobno orodje kot Composer. Razlika med njima je, da Bower uporabljamo pri upravljanju knjižnic za uporabniški vmesnik. Composer pa uporabljamo samo za upravljanje knjižnic v zalednem delu sistema. [7, 8]

4.4 MVC-arhitektura

Arhitektura Laravela temelji na osnovi MVC. Arhitektura MVC je način za izgradnjo aplikacije, ki vsebuje tri večje komponente.



Slika 4.1: Prikaz vzorca MVC [9].

Pri tej arhitekturi ima vsaka skupina komponent določeno nalogo. Modeli so razredi, ki skrbijo za manipulacijo s podatki v podatkovni bazi. Tipičen model je preslikava tabele iz podatkovne baze. Kontrolerji povezujejo modele in poglede. Pogledi so zadolženi za prikaz podatkov uporabniku. V pogledih imamo dostop do tistih podatkov, ki nam jih kontroler prej posreduje. [10]

4.5 Podatkovna baza MySQL

MySQL je sistem za upravljanje s podatkovnimi bazami [11]. Podatkovno bazo uporabljamo za shranjevanje vseh podatkov, ki jih v aplikaciji potrebujemo. Laravel nam olajša gradnjo in vzdrževanje podatkovne baze, saj omogoča migracije. Najprej v PHP-jeziku definiramo, katera polja bo določena tabela vsebovala, in nato z enim ukazom kreiramo celotno podatkovno bazo. Poleg tega lahko definiramo tudi sejalce tabel. To so datoteke, v katerih je zapis, na kakšen način in s katerimi podatki napolnimo tabele v bazi. Zaradi tega dvojega lahko z enim ukazom porušimo in ponovno postavimo ter s testnimi podatki napolnimo podatkovno bazo. Takšen način dela nam tudi olajša delo v skupinah, saj je ustvarjanje in spreminjanje tabel shranjeno v VCS. [12]

4.6 Bootstrap

Bootstrap je eno od najpopularnejših prosto dostopnih ogrodij za razvoj uporabniškega vmesnika. Uporablja se pri programiranju odzivnih spletnih strani [13]. Bootstrap smo uporabili pri implementaciji oblikovanja administracijske plošče in izgleda spletne strani, ki jo bo uporabljal uporabnik. Ogrodje olajša delo, saj definira določene razrede v CSS, ki omogočajo hitro uporabo in skladen videz po celi aplikaciji. [14]

4.7 JSON

JSON je preprost format za prenos podatkovnih modelov, ki so sestavljeni po principu množic. Namenjen je shranjevanju in izmenjavi podatkov. V našem primeru ga uporabljamo za komunikacijo med strežnikom in mobilno aplikacijo. Za JSON smo se odločili, ker ima lažjo sintakso od konkurenčnega XML-ja. Tudi čas porabljen za razčlenjevanje je krajši. [15]

Poglavje 5

Arhitektura sistema

V nadaljevanju smo opisali arhitekturo prototipa informacijskega sistema. Prototip temelji na arhitekturi odjemalec-strežnik. Prikazali smo njegove sestavne dele. Poleg tega bo tudi razvidno, kako se različne uporabljene tehnologije povezujejo v končni izdelek.

Prototip smo razdelili na štiri večje dele. Administracijsko ploščo, spletno aplikacijo za uporabnika, spletne servise in mobilno aplikacijo. Vsi deli razen mobilne aplikacije živijo na strežniku. Mobilna aplikacija komunicira s serverjem s pomočjo spletnih servisov. V prototipu nismo potrebovali narediti svoje rešitve za dogodke in galerijo, saj podatke pridobivamo iz Facebook Graph API-ja. Na ta način vzdrževalcem spletne aplikacije olajšamo delo, saj se podatki lahko naložijo le na Facebook.

Arhitektura sistema je predstavljena skladno s Kruchtenovim [16] arhitekturnim modelom 4+1. Pri predstavitvi smo uporabili pogled primerov uporabe, podatkovni pogled, procesni pogled ter implementacijski in fizični pogled, združena skupaj.

5.1 Primeri uporabe

Spletno aplikacijo lahko uporabljata dva različna tipa uporabnikov. Registrirani uporabniki, ki so lahko člani kluba, in neregistrirani uporabniki oz.

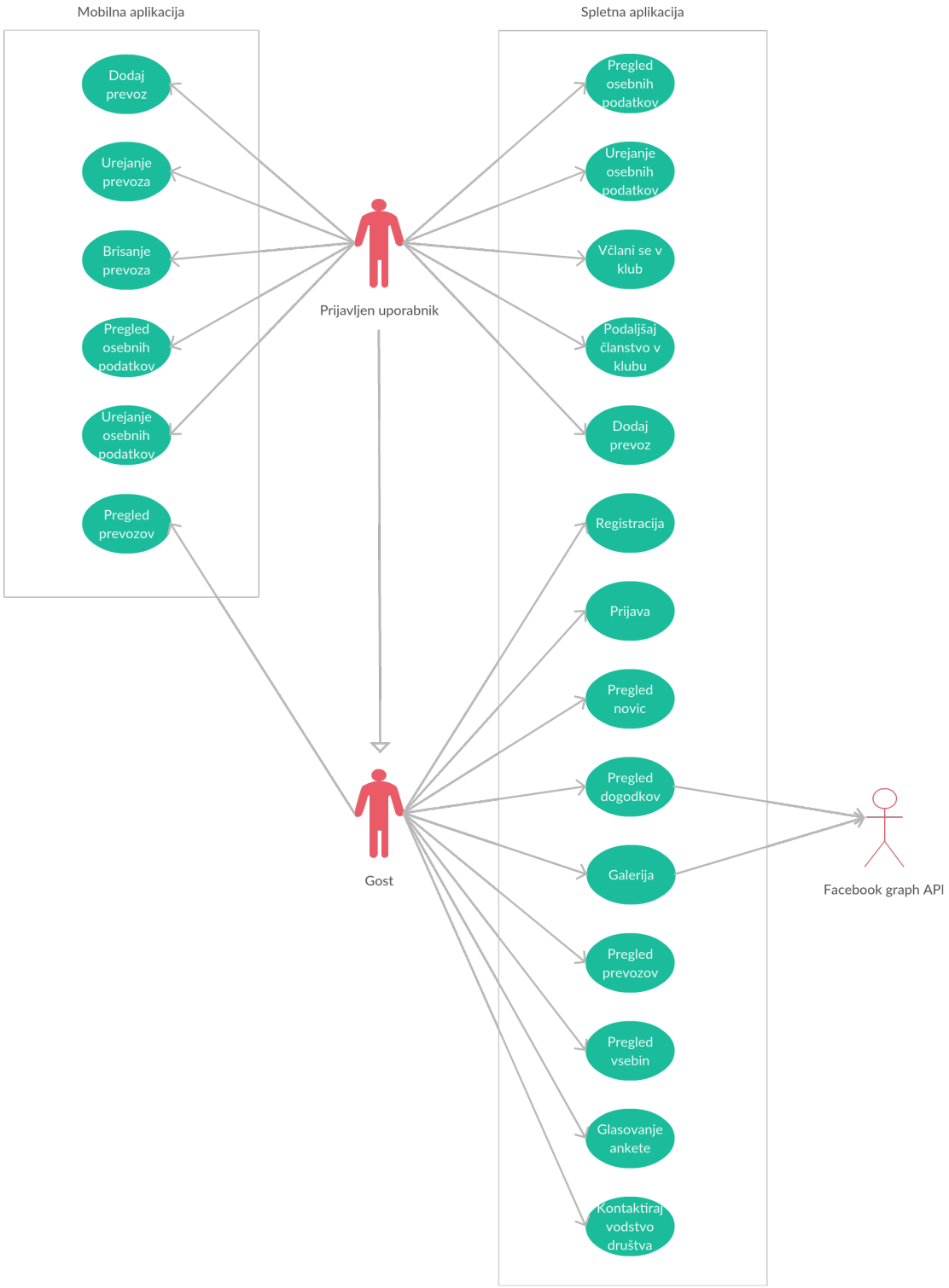
gostje spletne aplikacije. Vsak gost se lahko tudi registrira in nato včlani v klub. Prijavljen uporabnik in član kluba pa lahko podaljša svoje članstvo. Registriran uporabnik lahko v aplikaciji odda nov prevoz. Na Sliki 5.1 so vidni primeri uporabe spletne in mobilne aplikacije.

Poleg navadnih uporabnikov imamo tudi uporabnike s pravicami urejanja spletne aplikacije. Takšni uporabniki so razdeljeni v štiri večje skupine. Prevoznik ima pravice urejanja in brisanja prevozov vnesenih s strani drugih uporabnikov. Novinar lahko dodaja nove novice ali ureja ter briše obstoječe, poleg tega pa lahko tudi doda novo anketo. Moderator lahko ureja enake vsebine kot novinar in prevoznik, poleg tega pa lahko dodaja nove strani ali ureja in spreminja vsebino obstoječih. Administratorji imajo pravice urejanja vseh podatkov. Urejajo lahko enake podatke kot ostale tri skupine vključno z urejanjem članov in njihovih podatkov, kar pa ostale skupine ne morejo. Primeri uporabe administracijske plošče so vidni na Sliki 5.2.

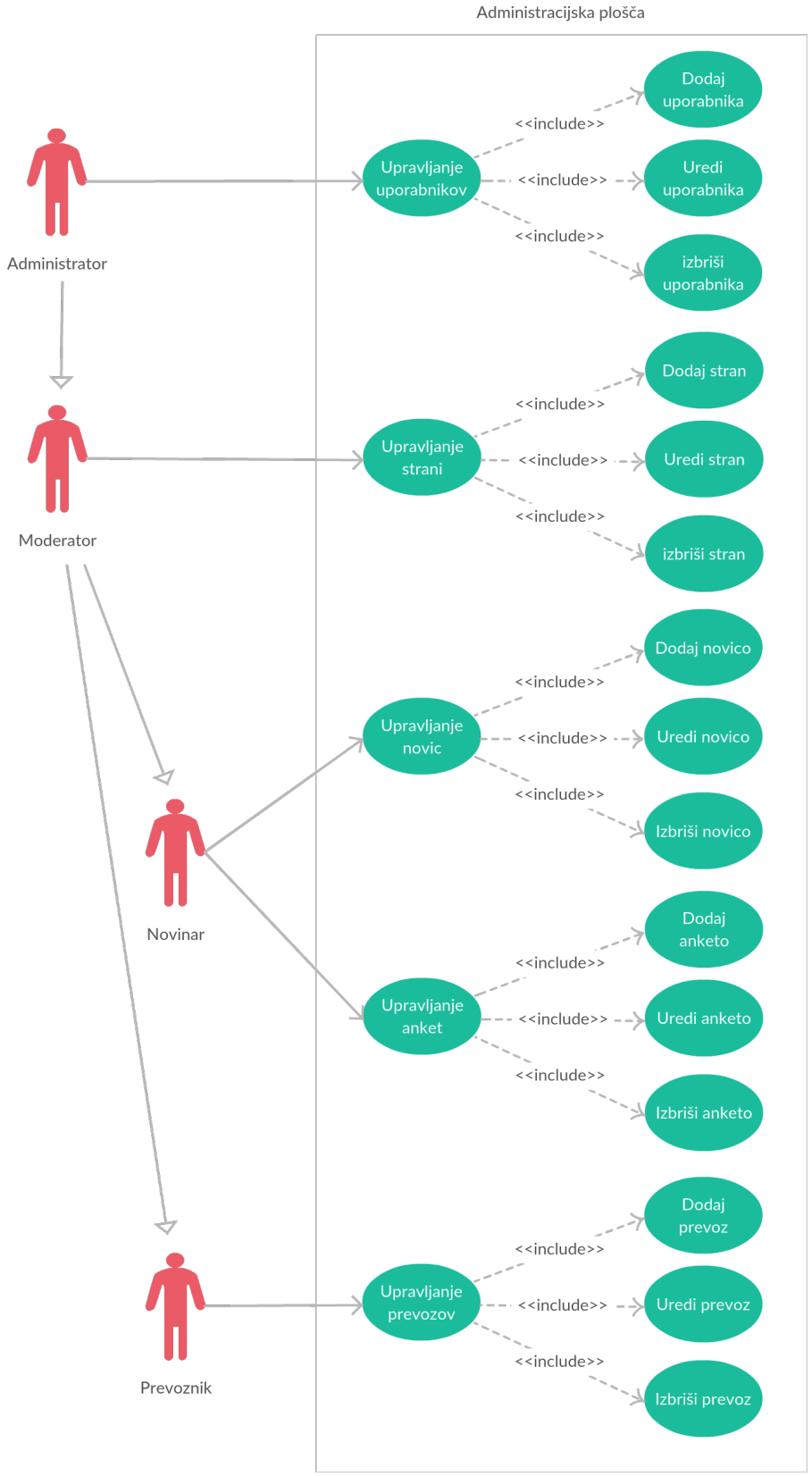
5.2 Podatkovni pogled

Podatkovna baza je relacijska in vsebuje 15 tabel. Sestavljena je iz tabel in relacij, ki hranijo podatke, ki so potrebni za delovanje spletne aplikacije (Slika 5.3).

V tabelah role_user, roles, permission_role in permissions se nahajajo vsi potrebni podatki za ločevanje uporabnikov oz. njihovih pravic. Povedo nam, kateri uporabniki lahko urejajo spletno aplikacijo in kaj smejo spreminjati oz. dodajati. V tabeli api_keys imamo shranjene vse oddane API-ključe, ki so potrebni, da mobilna aplikacija lahko komunicira s strežnikom. Dnevniki API-zahtevkov pa so shranjeni v api_logs. Iz dnevnikov je razvidno, koliko zahtevkov na strežnik pošlje posamezen uporabnik. Uporabnika se blokira, če je presegel dnevni limit zahtevkov ali če je bil zaznan kakršen koli napad na strežnik s strani tega uporabnika. Tabela news vsebuje novice, ki jih je vnesla oseba s pravicami. Poleg tega ima tudi povezavo na photos, ki vsebuje podatke o slikah dodanih k določeni novici. Tabeli survey_questions



Slika 5.1: Diagram primerov uporabe spletne in mobilne aplikacije.



Slika 5.2: Diagram primerov uporabe administracijske plošče.

in survey_answers se uporabljata za shranjevanje anket, tabela transports pa za shranjevanje vnesenih prevozov. Pages se uporablja za shranjevanje vsebin oz. posameznih strani. Poleg tabele users, ki vsebuje podatke o registriranih uporabnikih, imamo še tabelo password_resets, ki vsebuje podatke, potrebne za uporabnike, ki so pozabili svoje geslo in ga želijo resetirati.

Tabelo migrations pa uporablja ogrodje Laravel. V njej beleži, katere migracije so se že izvedle. Tako ve, katere migracije še mora izvesti oz. katere migracije so že bile izvedene in jih mora povrniti v stanje pred spremembo.

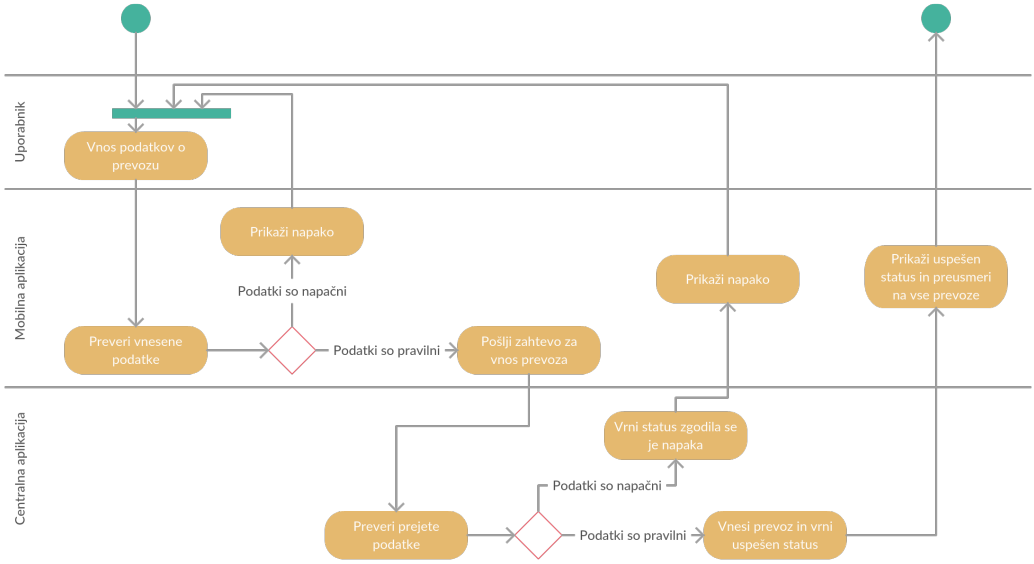
5.3 Procesni pogled

Procesni pogled prikaže in opiše aktivnosti sistema in komunikacijo med njimi. Kot priročen se izkaže, kadar imamo več vzporednih aktivnosti. V procesnem pogledu smo obravnavali le procesno bolj kompleksne dele sistema, in sicer: prijava, pregled dogodkov, prikaz in izpolnjevanje ankete, vnos prevoza na mobilni napravi. V delu smo prikazali le aktivnost vnosa prevoza.

Na sliki 5.4 je prikazana aktivnost vnosa novega prevoza s strani že prijavljenega uporabnika. Pri izvedbi procesa imamo tri ključne elemente. Uporabnika, ki v sistem s pomočjo mobilne aplikacije želi vnesti nov prevoz. Mobilno aplikacijo, ki je vmesnik med uporabnikom in centralnim strežnikom. Aplikacija skrbi za prenos podatkov med obema entitetama. Centralna aplikacija prejema zahtevke od mobilne aplikacije. Naredi zahtevano in nato pošlje odgovor nazaj mobilni aplikaciji.

Aktivnost se začne z uporabnikom, ki želi vnesti nov prevoz v sistem. Uporabnik v mobilni aplikaciji vnese podatke prevoza in potrdi vnos. Aplikacija nato preveri podatke in vrne napako, če so podatki napačni. Če so podatki pravilni, pošlje zahtevek na strežnik za vnos prevoza. Strežnik ponovno preveri podatke in vrne status »napaka« mobilni aplikaciji, če so podatki napačni. Če so ti pravilni, nadaljuje z vnosom prevoza in vrne uspešen status. Mobilna aplikacija status prikaže in uporabnika preusmeri na vse

Slika 5.3: ER-diagram podatkovne baze.



Slika 5.4: Procesni pogled vnosa prevoza na mobilni aplikaciji.

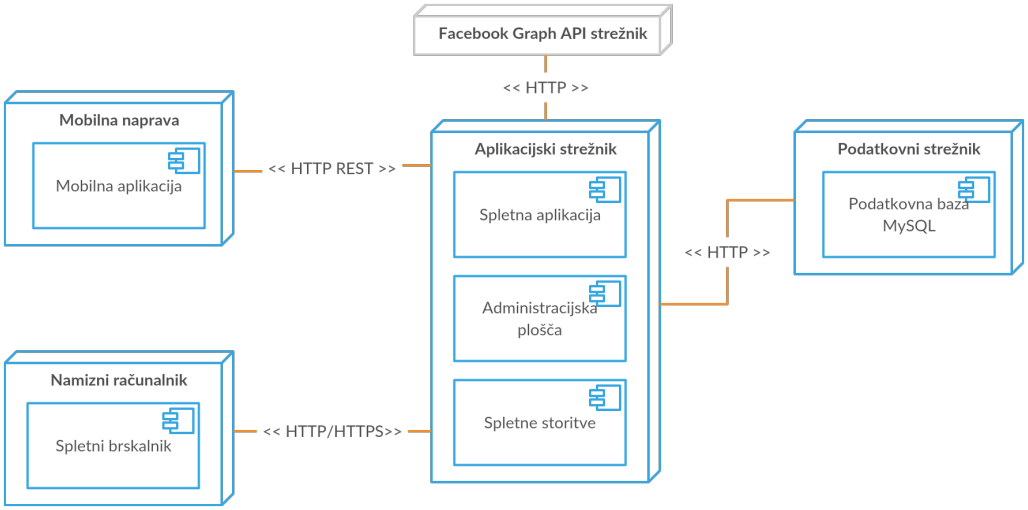
prevoze.

Slika 5.4 nazorno prikazuje tipično komunikacijo med mobilno napravo in strežnikom. Pri prikazu aktivnosti iz spletne aplikacije v procesnem pogledu izločimo mobilno napravo. Vse podatke, torej celotno spletno stran, ki se prikaže pri uporabniku, pošljemo s strežnika in prikažemo v brskalniku.

5.4 Implementacijski in fizični pogled

Implementacijski pogled prikaže sistemske komponente, knjižnice, aplikacije ... Fizični pogled prikaže postavitev ter povezavo strežnikov in naprav v sistemu (Slika 5.5).

Uporabnik lahko dostopa do podatkov z računalnikom ali mobilno napravo. Aplikacijski stežnik komunicira s podatkovnim strežnikom. Poleg tega komunicira tudi s strežnikom Facebook Graph API, ki pa ni v naši domeni.



Slika 5.5: Komponentni diagram sistema v povezavi z vozlišči.

Poglavje 6

Razvoj prototipa

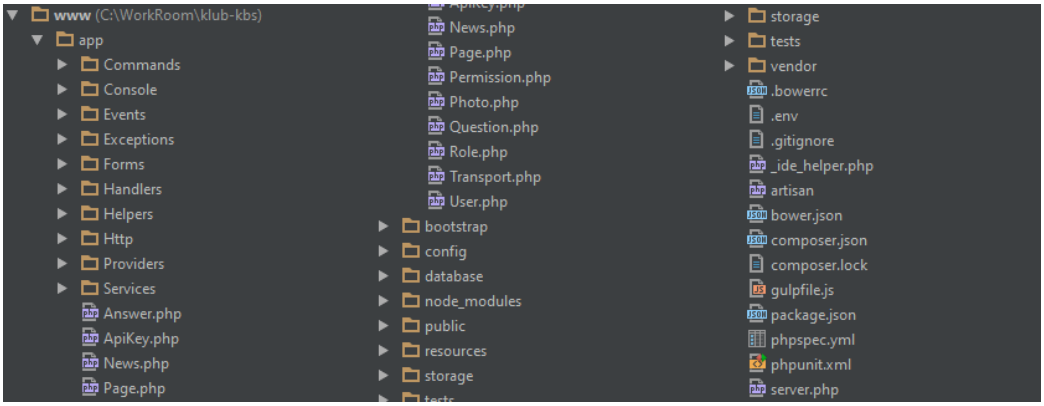
6.1 Strežniški del

Naloga strežnika je kontrolirati in omogočiti dostop do njegovih sredstev. Poleg tega je naloga strežnika komunikacija s stranjo odjemalca s pomočjo spletnih servisov. Aplikacijo, ki živi na strežniku, smo napisali s priljubljenim ogrodjem Laravel. Izgled mobilne aplikacije in spletne aplikacije sta prispevala člana Kluba belokranjskih študentov.

6.1.1 Podatkovni strežnik

Za lokalni razvoj smo si postavili Laravel Homestead. Homestead je uradno predpripravljen Vagrant box, ki ima že vnaprej nameščena in pripravljena orodja, potrebna pri razvoju spletnih aplikacij. Razvijalcu več ni potrebno namestiti PHP, HHVM, MySQL, spletnega strežnika ali katerega koli drugega orodja na lokalno napravo. Ker lahko Vagrant box na hitro uničimo in postavimo na novo, ne rabimo več skrbeti, da bi pri razvoju kakor koli škodovali našemu lokalnemu sistemu.

Preden smo lahko zagnali Homestead, smo morali namestiti VirtualBox in Vagrant. VirtualBox je programski paket, ki omogoča pripravo in uporabo navideznih računalnikov. Vagrant pa je orodje za hitro ustvarjanje navideznih računalnikov, ki so namenjeni razvojnim okoljem.



Slika 6.1: Datotečna struktura spletne aplikacije.

6.1.2 Datotečna struktura projekta

Mapa z imenom app vsebuje jedro naše kode. V njej se nahaja vsa logika potrebna za normalno delovanje spletne aplikacije. V mapi config se nahaja vsa konfiguracija za aplikacijo. V njej lahko najdemo nastavitev jezika in tudi nastavitev, katera vrsta podatkovne baze naj se uporablja. V mapi database imamo migracije in sejalce tabel. Migracije so datoteke, v katerih definiramo strukturo posamezne tabele oz. spremembe nad tabelami v podatkovni bazi. V sejalcih pa definiramo, s katerimi podatki in na kakšen način naj se tabele napolnijo. Mapa public je edina, ki je navzven vidna uporabniku. V njej imamo slike, stile in vse, kar bomo potrebovali za uporabniški vmesnik. V mapi resources imamo datoteke scss, v katerih definiramo stilske datoteke. V njej imamo tudi mapo s pogledi in večjezične definicije. V mapi s pogledi so definirani vsi pogledi za našo aplikacijo. Večjezične definicije uporabljamo takrat, kadar imamo potrebo po več jezični aplikaciji.

Najpomembnejše stvari se nahajajo v mapi app. V njej imamo vse kontrolerje, forme, evente, modele itd. Na sliki 6.1 vidimo, da imamo datoteke z imeni User, Answer, ApiKey, News ... To so naši modeli, ki jih uporabljamo pri shranjevanju in pridobivanju podatkov iz baze. V mapi Http se poleg druge kode nahajajo tudi kontrolerji, ki so zadolženi za komunikacijo s pogledi in modeli. Imamo pa tudi datoteko routes.php, v kateri so definirane

akcije, ki se izvedejo, kadar obiščemo določen URL-naslov:

```
Route::get('/prevozi', ['as' => 'app.transports', 'uses' =>
    'App\\TransportsController@index']);
```

V zgornjem primeru lahko vidimo, da se izvede metoda index iz kontrolerja TransportsController, če bo uporabnik obiskal pot `http://www.klub-kbs.si/prevozi`. V metodi index s pomočjo modela Transport pridobimo vse aktualne prevoze iz baze.

Če želimo dodati nov vnos v tabelo, pošljemo zahtevek post.

```
Route::post('/users', ['as' => 'admin.users.store', 'uses' =>
    'Admin\\UsersController@store']);
```

Zgornji primer se izvede, kadar želi administrator dodati novega uporabnika v sistem. Tukaj dobimo zahtevek post, nato pa se izvede metoda store v kontrolerju UsersController. V metodi nato preverimo podatke, ki so prispeli v jedru zahtevka. Če so podatki pravilni, s pomočjo modela User vnesemo novega uporabnika v bazo.

6.1.3 Spletni servisi

Spletni servisi [17] so standardiziran način komunikacije med dvema napravama v omrežju. Za komunikacijo lahko uporabljamo JSON, XML, SOAP ... Na serverju definiramo dostopne točke, ki jih naprava uporabnika nato pokliče, kadar želi izvesti določeno akcijo. Opisi dostopnih točk se nahajajo na naslovu `http://klub-kbs.si/api/v1`. Na koncu URL-naslova je oznaka različice, s katero z lahkoto sledimo spremembam servisa. Če bi uporabljali api brez različice, bi lahko ob spremembah onesposobili delovanje aplikacij, ki so že nameščene na napravah.

Neregistrirani uporabniki imajo dostop do prijave, registracije in pregleda vseh aktualnih prevozov. To lahko vidimo pri opisih dostopnih točk. Tiste dostopne točke, ki imajo lastnost use authorization nastavljene na false, lahko

uporabljamo brez predhodne prijave uporabnika. Za vse ostale se moramo prej prijaviti ali registrirati.

Z mobilno aplikacijo uporabnika registriramo tako, da pošljemo zahtevek post na točko register.

```
{
    "url": "http://localhost:8000/api/v1/register",
    "method": "POST",
    "parameters": "name, surname, email, password, phone",
    "description": "Register new user",
    "use authorization": false
}
```

Če uporabnik s tem e-poštnim naslovom še ne obstaja, dobimo od strežnika odgovor v JSON-obliki. Odgovor vsebuje podatke registriranega uporabnika in API-ključ, ki ga potem mobilna aplikacija uporablja pri komuniciranju s strežnikom. Na ta način strežnik ob naslednjih zahtevkih ve, s katerim uporabnikom komunicira.

```
{
    "data": {
        "name": "John",
        "surname": "Doe",
        "email": "john.doe@gmail.com",
        "phone": null,
        "slug": "john-doe",
        "api_key": "e0cfe359a2ed5f1f6892e17f82ef7305727d9174"
    }
}
```

API-ključ uporabljamo v glavi vsakega zahtevka na strežniku. Zapišemo ga kot vrednost v parameter X-Authorization.

Nov prevoz lahko oddajajo samo registrirani uporabniki. Ob oddaji novega prevoza aplikacija pošlje zahtevek post na točko transports z dodanimi podatki, potrebnimi za shranjevanje novega prevoza.

V primeru, da strežnik ne dobi pravilnega API-ključa v glavi zahtevka, odgovori z napako. Prevoz je vnesen uspešno, kadar je API-ključ pravilen in se postopek vnosa zgodi brez napak.

```
{
    "error": {
        "code": "GEN-UNAUTHORIZED",
        "http_code": 401,
        "message": "Unauthorized"
    }
}
```

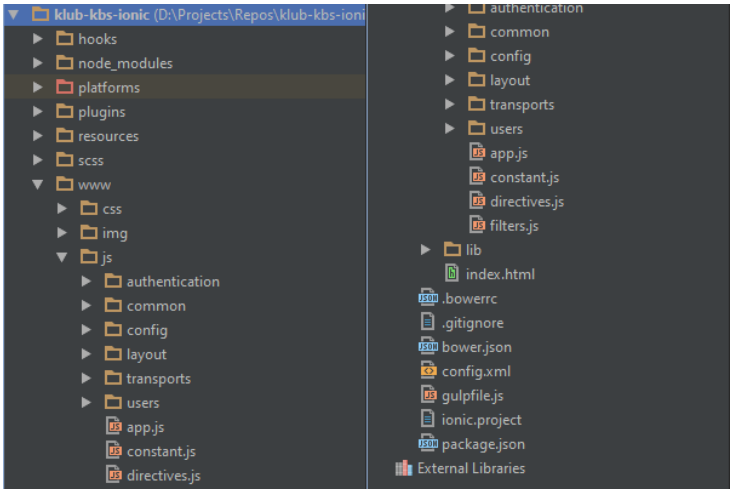
6.2 Mobilni del

Mobilna aplikacija je vmesni člen med strežnikom in končnim uporabnikom. Napisati smo se jo odločili s pomočjo ogrodja Ionic. Omogoča nam uporabo že poznanih tehnologij HTML, CSS in JavaScript. Prednost tega je, da lahko iz ene izvorne kode zgradimo aplikacijo za operacijske sisteme Android, IOS ali Windows Phone.

6.2.1 Datotečna struktura projekta

V mapi www se nahaja jedro naše aplikacije. Vsa logika se nahaja v mapi js. Datoteke za posamezno funkcionalnost imamo v skupni mapi. Na ta način si zagotovimo hitro iskanje potrebnih datotek, ko jih potrebujemo (Slika 6.2).

Kodo imamo razdeljeno na tri večje nivoje. Servisi so najnižji del, ki je zadolžen za komunikacijo s strežnikom ter shranjevanje in branje podatkov iz lokalnega pomnilnika. Drugi nivo so kontrolerji, ki so vmesni člen med



Slika 6.2: Arhitektura mobilne aplikacije.

servisi in pogledi. Najvišji nivo pa predstavljajo pogledi, s katerimi uporabnik upravlja aplikacijo.

6.2.2 Avtentikacija

V datoteki authentication.routes.js smo definirali kontroler in pogled, ki naj se izvedeta, ko obiščemo določeno stran. Ko obiščemo /login, se naloži pogled _login.html in kontroler LoginCtrl.js. Ukaz restrictAccess: ['notLogged'] nam pove, da se ta stran prikaže samo neprijavljenim uporabnikom.

```
.state('login', {
    url: '/login',
    templateUrl: 'js/authentication/_login.html',
    controller: 'LoginCtrl',
    data: {
        restrictAccess: ['notLogged']
    }
})
```

Za določene zahtevke na stežnik mora biti uporabnik prijavljen, zato

imamo prestreznik, ki ob vsakem zahtevku na strežnik doda API-ključ v glavo zahtevka, če le-ta obstaja. API-ključ smo dobili kot odgovor na uspešno prijavo uporabnika.

6.2.3 Prikaz prevozov

Ob prihodu na vse prevoze se naloži pogled /transports/_all.html in kontroler transports.ctrl.js. Preden se kontroler popolnoma naloži, pokliče funkcijo getAll v servisu namenjenem za prevoze.

```
function getAll() {
    if (cachedTransports) {
        return $q.when(angular.copy(cachedTransports));
    } else {
        return $http.get(Config.backendUrl + 'transports')
                .then(function (res) {
            cachedTransports = res.data.data;
            return angular.copy(cachedTransports);
        });
    }
}
```

Prevozi se pridobijo iz strežnika ali iz predpomnilnika na napravi. Kontroler dobljene podatke posreduje naprej pogledu, ki je zadolžen za prikaz podatkov.

Poglavje 7

Predstavitev funkcionalnosti

V nadaljevanju smo prikazali funkcionalnosti spletne aplikacije za uporabnika, administracijske plošče in mobilne aplikacije. Vsi prikazani podatki so testni.

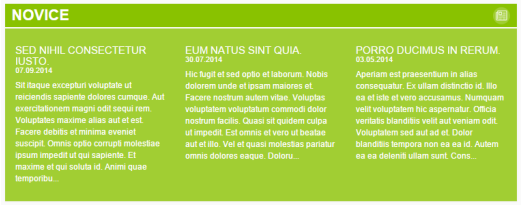
7.1 Spletna aplikacija za uporabnika

Ob obisku aplikacije se uporabniku na začetni strani prikažejo zadnje 3 novice (Slika 7.1a), aktualni prevozi (Slika 7.1b), prihajajoči dogodki, pridobljeni z Facebooka (Slika 7.2a) ter aktualna anketa (Slika 7.2b).

Uporabnik ima vpogled v vse novice ter v vsako posebej. Lahko si ogleda vse aktualne prevoze in galerijo, ki vsebuje slike, posnete na dogodkih, organiziranih s strani kluba. Prav tako ima tudi možnost pregleda projektov, izdelanih s strani članov kluba. Lahko se tudi registrira in prijavi v aplikacijo. S prijavo ima možnost postati član društva ali podaljšati članstvo. Lahko odda tudi nov prevoz.

7.2 Administracijska plošča

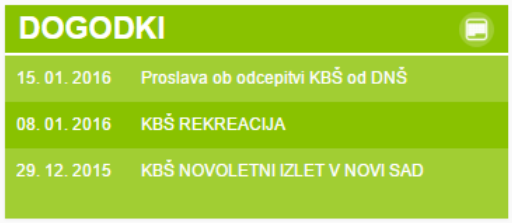
Pri implementaciji administracijske plošče smo uporabili odprtokodno knjižnico za pomoč pri izgledu administracijskih plošč z imenom AdminLTE. Knjižnica



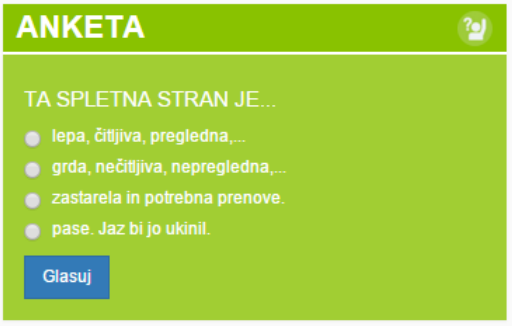
(a) Zadnje tri novice.

(b) Aktualni prevozi.

Slika 7.1: Novice in prevozi na prvi strani.

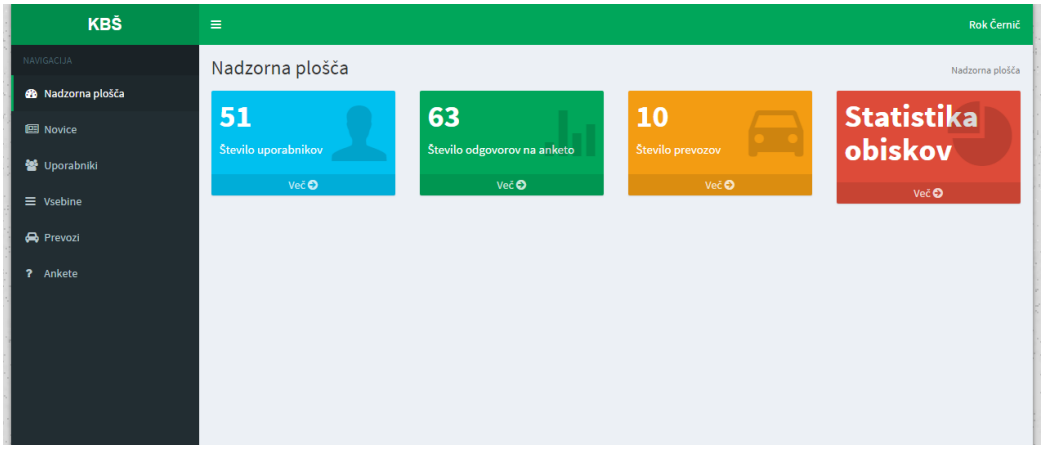


(a) Prihajajoči dogodki.



(b) Integrirana anketa.

Slika 7.2: Dogodki in anketa na prvi strani.



Slika 7.3: Nadzorna plošča.

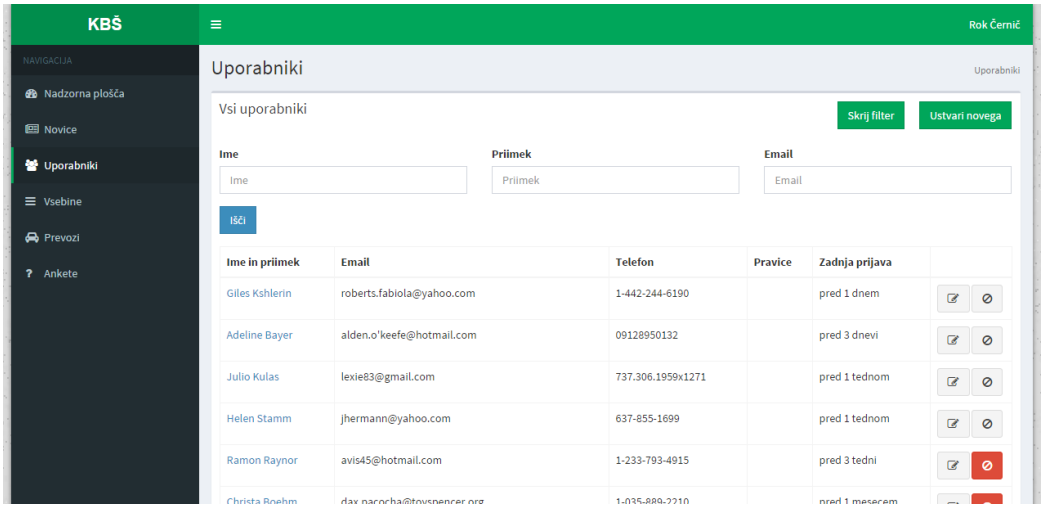
vnaprej definira izgled in uporabi različne dodatne knjižnice za prikaz grafov, urejevalnikov teksta, animacij itd.

Ob prijavi v administracijski del aplikacije se najprej odpre nadzorna plošča (Slika 7.3). V njej imamo prikazane najpomembnejše podatke za administratorja. Prikažemo število registriranih uporabnikov, število odgovorov na anketo, število aktualnih prevozov itd. Zraven je tudi povezava do sistema Google Analytics, kjer lahko vidimo statistiko obiskov spletne aplikacije. Za nadgradnjo imamo namen podatke pridobiti iz Google Analytics in jih prikazati v nadzorni plošči. Pri rezultatih ankete imamo prav tako namen narediti podrobnejšo analizo posamezne ankete.

Uporabnike s pravicami urejanja aplikacije delimo na 4 skupine: novinar, prevoznik, moderator in administrator. Pravice urejanja za vsako skupino so vidne na sliki 5.2.

Upravljanje podatkov v aplikaciji je narejeno na podoben način, zato bomo v nadaljevanju prikazali le upravljanje uporabnikov, npr. dodajanja novega, urejanje obstoječega in brisanje starega uporabnika.

Administrator ima pregled nad vsemi registriranimi uporabniki (Slika 7.4). Lahko jih tudi filtrira po imenu, priimku ali elektronskemu naslovu. Posameznega uporabnika lahko tudi blokira in mu onemogoči prijavo v profil. Do

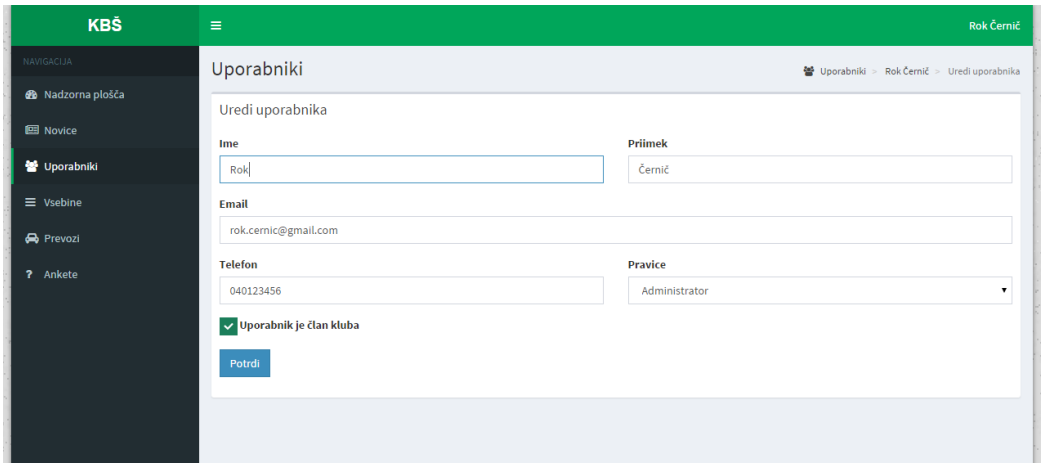


Slika 7.4: Predogled vseh uporabnikov.

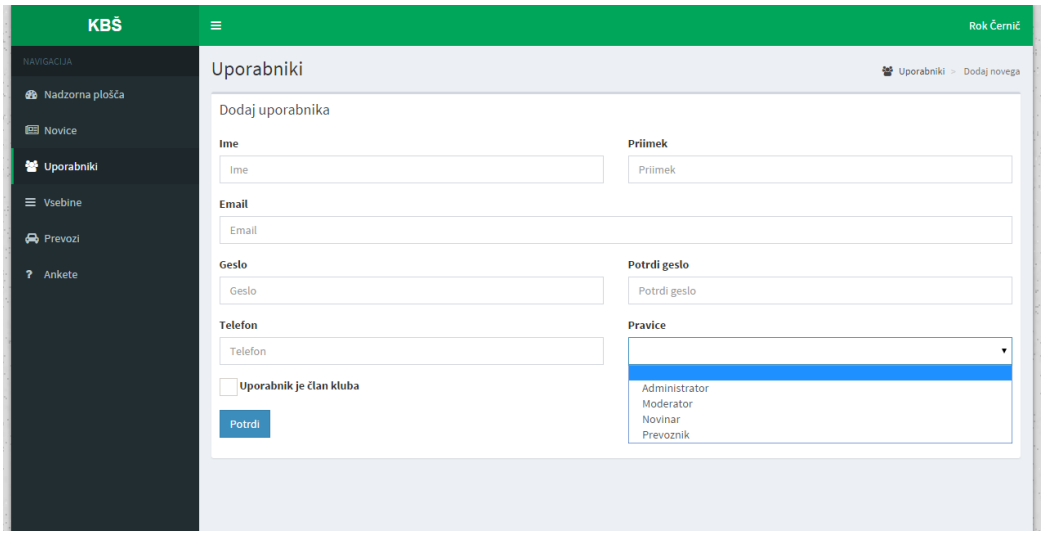
podrobnosti dostopa tako, da pritisne na ime in priimek. V podrobnostih ima možnost urejanja ali izbris uporabnika. Če izbere urejanje se mu odpre obrazec za urejanje (Slika 7.5). Lahko pa tudi doda novega uporabnika. V tem primeru se mu odpre obrazec za dodajanje (Slika 7.6). Uporabniku lahko dodeli katero koli od obstoječih pravic in pa tudi označi, ali je včlanjen v klub.

7.3 Mobilna aplikacija

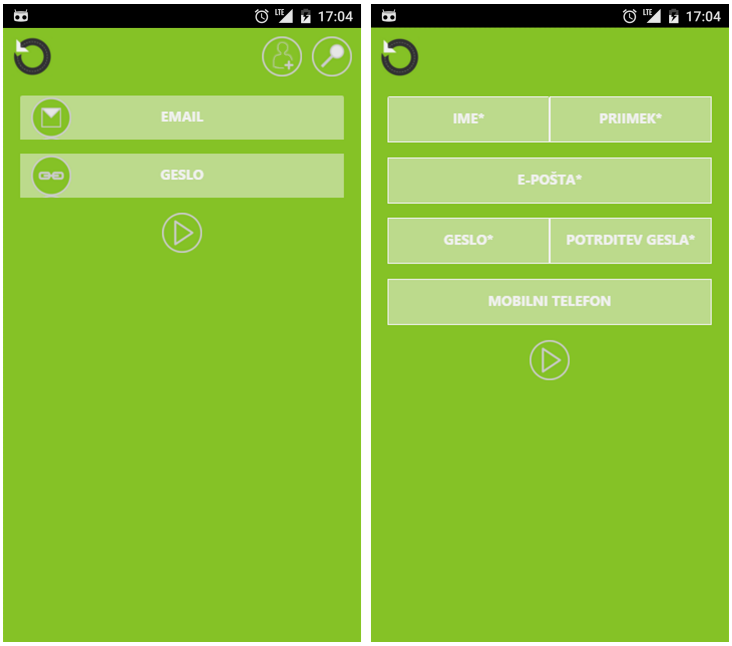
Razširjenost pametnih telefonov je iz dneva v dan večja, saj postajajo vse cenejši, njihove zmogljivosti pa vse večje. Mobilne aplikacije pa postajajo vse bolj priljubljene, saj so uporabniku na voljo, ko jih le-ta potrebuje. Mobilna aplikacija je program, ki se izvaja na pametnih telefonih, tablicah ali mobilnih napravah [18]. Odločili smo se, da ponudimo mobilno aplikacijo, v kateri bo lahko študent videl aktualne prevoze, poleg tega pa oddal tudi svojega. Aplikacija komunicira s strežnikom s pomočjo zgoraj opisanih spletnih servisov. Primeri uporabe mobilne aplikacije so vidni v diagramu 5.1.



Slika 7.5: Obrazec za urejanje uporabnika.



Slika 7.6: Obrazec za dodajanje novega uporabnika.



Slika 7.7: Obrazca za prijavo in registracijo.

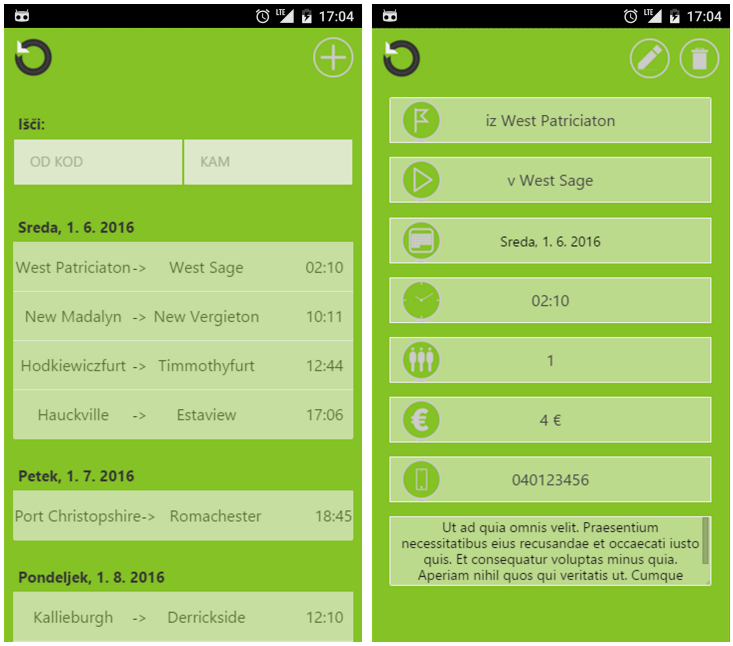
7.3.1 Avtentikacija

Uporabnik, ki želi oddati prevoz, se mora najprej registrirati oz. prijaviti v aplikacijo (Slika 7.7). Ko uporabnik izpolni obrazec za prijavo, se ta pošlje na strežnik. Če je bila prijava uspešna, dobi v jedru odgovora API-ključ, ki ga uporabi pri naslednjih zahtevah na strežnik.

7.3.2 Prevozi

Ena izmed ključnih funkcionalnosti aplikacije je pregled vseh prevozov in dodajanje novih (Slika 7.8). Ob prvi potrebi po prevozih se izvede zahteva na strežnik. Strežnik odgovori z listo vseh prevozov. Aplikacije dobljene prevoze shrani v začasni pomnilnik naprave. S tem zagotovimo napravi hitrejši dostop do podatkov. S pomočjo posebne akcije lahko uporabnik kadar koli zahteva ponoven prenos prevozov s strežnika.

Uporabnik lahko vidi vse prevoze v skupinah po dnevih. Prevozi, ki niso



Slika 7.8: Prikaz vseh prevozov in enega specifičnega.

več aktualni, niso prikazani. S klikom na prevoz vidimo več podrobnosti. Če uporabnik želi oddati nov prevoz se, mora najprej registrirati ali prijaviti.

7.3.3 Namestitev aplikacije

Aplikacija bo na voljo na portalu Google Play. Dostopna bo vsem uporabnikom mobilnih naprav z operacijskim sistemom Android različice vsaj 4.0. To je zahteva ogrodja Ionic. V uradni dokumentaciji je sicer zapisano, da naj bi aplikacija delovala tudi na napravah od različice 2.3 naprej. Ob namestitvi moramo potrditi pravice za uporabo interneta.

Poglavje 8

Zaključek

V sklopu diplomskega dela smo naredili analizo informacijskih sistemov za podporo študentskim klubom po Sloveniji. S pomočjo analize smo z upravnim odborom izdelali anketo, s katero smo analizirali potrebe bodočih uporabnikov. Iz analize potreb smo dobili najpomembnejše funkcionalnosti za uporabnike. Izbrane funkcionalnosti smo implementirali v izdelan prototip. Pri izdelavi prototipa smo uporabili le odprtokodne rešitve.

Strežniški del smo izdelali s pomočjo ogrodja za izdelavo spletnih aplikacij. Razdelili smo ga na administracijsko ploščo, ki omogoča uporabnikom s pravicami urejanje vsebin, aplikacijo za uporabnika, ki omogoča pregled novic, dogodkov, prevozov, slik, in na spletni servis, ki omogoča komunikacijo z mobilnim delom. Spletno aplikacijo smo povezali tudi s portalom Facebook. S tem smo zagotovili lažje urejanje vsebin.

Mobilna aplikacija ponuja uporabniku priročen pregled aktualnih prevozov. Uporabnik se lahko tudi registrira in vnese svoj prevoz. Z izdelavo mobilne aplikacije smo omogočili uporabnikom dostop do prevozov, kadar jih le-ti potrebujejo.

Pri razvoju prototipa smo se držali principov dobrega programiranja in gradnje aplikacij. S tem smo si zagotovili hitro in lažje nadgrajevanje ter dodajanje novih funkcionalnosti že obstoječemu prototipu. Poleg tega je manjša možnost pojavljanja napak v kodi. V naslednji različici bi lahko

omogočili uporabniku nalaganje potrdil o šolstvu, prikaz slik na prvi strani z Instagrama. Administratorjem bi lahko omogočili tudi pošiljanje elektronske pošte določenim članom ali celotnim skupinam. Lahko bi implementirali Google Analytics. Gre za Googlovo storitev za analizo obiskov spletnih strani. Do teh podatkov bi imeli dostop le administratorji. Podatki pa bi jim prikazali, kateri članki oz. vsebine na strani so najbolj aktualne.

S prototipom smo želeli izboljšati delovanje kluba, upravnemu odboru olajšati delo ter članom omogočiti hitrejše pridobivanje podatkov pomembnih pri študiju in življenju izven njega. V aplikacijo bi lahko dodali prikaz najbližjih prevozov in na ta način olajšali uporabniku iskanje prevoza. V času razvoja prototipa so aplikacijo testirali člani upravnega odbora kluba, ki prav tako študenti. Z njihovo pomočjo smo hitreje odkrili obstoječe napake in jih tako tudi popravili.

Literatura

[1] R. Johns. *Likert Items and Scales*, 2010, str. 2.

[2] Harry N. Boone, Deborah A. Boone, "Journal of Extension", *Analyzing Likert Data*, št. 50, zv. 2, str. 1–3, 2012.

[3] J. Miller, P. Haden. *Statistical Analysis with The General Linear Model*, 2006, str. 9–12.

[4] (2015) The Best PHP Framework for 2015. Dosegljivo na: http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/. [Dostopano 17. 12. 2015]

[5] J. Vo. *Learning Laravel: The Easiest Way*, 2014, str. 6–7.

[6] (2014) 5 Best Mobile Web App Frameworks. Dosegljivo na: http://moduscreate.com/5-best-mobile-web-app-frameworks-ionic-angulalrjs/. [Dostopano 25. 12. 2015]

[7] (2013) What is PHP Composer?. Dosegljivo na: http://culttt.com/2013/01/07/what-is-php-composer/. [Dostopano 10. 12. 2015]

[8] (2014) Manage Project Dependencies with Bower and Composer. Dosegljivo na: http://techportal.inviqa.com/2014/01/29/manage-project-dependencies-with-bower-and-composer/. [Dostopano 5. 12. 2015]

[9] S. Casteleyn, F. Daniel, P. Dolog, M. Matera. *Engineering Web Applications*, 2009, str. 228–229.

[10] (2015) Model-view-controller. Dosegljivo na: https://en.wikipedia.org/wiki/Model-view-controller. [Dostopano 9. 1. 2016]

[11] A. Butcher. *Sams Teach Yourself MySQL in 21 Days (2nd Edition)*, 2002, str. 9.

[12] Migrations & Seeding. Dosegljivo na: https://laravel.com/docs/5.0/migrations. [Dostopano 9. 1. 2016]

[13] (2014) The 5 Most Popular Frontend Frameworks of 2014 Compared. Dosegljivo na: http://www.sitepoint.com/5-most-popular-frontend-frameworks-compared/. [Dostopano 26. 12. 2015]

[14] (2016) Bootstrap (front-end framework). Dosegljivo na: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework). [Dostopano 26. 12. 2015]

[15] (2016) JSON. Dosegljivo na: https://en.wikipedia.org/wiki/JSON. [Dostopano 27. 12. 2015]

[16] P. Kruchten, "Architectural Blueprints", *The "4+1" View Model of Software Architecture*, št. 12, zv. 6, str. 1–2, 1995.

[17] Web services. Dosegljivo na: http://www.webopedia.com/TERM/W/Web_Services.html. [Dostopano 10. 1. 2016]

[18] (2014) Mobile app. Dosegljivo na: http://en.wikipedia.org/wiki/Mobile_app. [Dostopano 11. 1. 2016]